Content-Access QoS in Peer-to-Peer Networks Using a Fast MDS Erasure Code

Laurent Dairaine[†,*], Jérôme Lacan[†], Laurent Lancérica[†,‡], Jérôme Fimes[†,‡]

Affiliation
[†]ENSICA, 1 place Émile Blouin, 31056 Toulouse cedex 5, France
[*]LAAS-CNRS, 5 avenue du Colonel Roche, 31077, Toulouse cedex 4, France
[‡]TéSA, 2, rue Charles Camichel, F-31071 Toulouse, Cedex 7, BP 7122, France

Emails
{Laurent.Dairaine, Jerome.Lacan, Laurent.Lancerica, Jerome.Fimes}@ensica.fr

Corresponding author
Laurent Dairaine
Laurent.Dairaine@ensica.fr
ENSICA, 1 place Emile Blouin, 31056 Toulouse cedex 5, France
Tel. +33 1 61 61 86 83
Fax. +33 1 61 61 86 88

Abstract
This paper describes an enhancement of content access Quality of Service in peer to peer (P2P) networks. The main idea is to use an erasure code to distribute the information over the peers. This distribution increases the users' choice on disseminated encoded data and therefore statistically enhances the overall throughput of the transfer. A performance evaluation based on an original model using the results of a measurement campaign of sequential and parallel downloads in a real P2P network over Internet is presented. Based on a bandwidth distribution, statistical content-access QoS are guaranteed in function of both the content replication level in the network and the file dissemination strategies. A simple application in the context of media streaming is proposed. Finally, the constraints on the erasure code related to the proposed system are analysed and a new fast MDS erasure code is proposed, implemented and evaluated.

Keywords
Peer-to-Peer, Quality of Service, Data Storage, Performance Evaluation, Erasure Code.

1. INTRODUCTION

The popularity of Peer-to-Peer (P2P) file sharing applications offers new prospects to Internet end-users. In this context, users can move from the simple consumer state to the active state of publisher, sharing their contents through the network. P2P network is a high logical level network architecture built over end-user nodes interconnected by a classical computer network infrastructure such as Internet. Main P2P systems provide services for storing, finding and downloading data. The main characteristic of P2P networks is to avoid any centralized point, thus allowing the building of new distributed services which do not rely on asymmetric models such as master-slave, consumer-supplier or client-server.

P2P networks feature an enhancement of the use of information, bandwidth, and computing resources [1]. The classical client-server model reduces the information resource usage since it makes difficult to find and retrieve data from centralized servers. A decentralized solution with a collaboration of peers would avoid those difficulties. Indeed, central servers which attract a heavy load of requests introduce a bottleneck bandwidth at network and system level, leading to the building of large web farms and broadband network resources. The dissemination of information over a set of peers allows a natural distribution of the load over the underlying network and the end-systems. Network resources use could also be enhanced introducing high level routing functions in the P2P system. It would then speed up data access with the use of load balancing or any other approaches. Finally, the processing and the storage of resources are shared over the P2P network: e.g., a file can be replicated over a large number of peers improving its availability, fault tolerance and Quality of Service (QoS) in distributed systems [2].

However, in most of P2P networks [3], [4], [5], each peer is an autonomous and independent system. The P2P network can be then considered as a heterogeneous and variable network topology composed of nodes whose lifetime and access bandwidth may vary. A detailed observation of the Gnutella and Napster network is provided in [6]. As a result, numerous research level issues are related to the conception of such systems. Scalability, authentication, routing, fault tolerance, data localization and data access are examples of hot topics concerning the P2P domain. The contribution of this paper concerns the last issue in order to evaluate and provide statistical guarantees on data access performance for several file dissemination strategies.

Indeed, the data access performance strongly depends on the file dissemination in the network. A classical approach consists in enhancing the localization of replicated copies [7], [8]. The end user then downloads one of them from the closest peer. Typical enhancement of this scheme consists in getting various blocks from different peers in parallel. This method is used by a majority of classical file sharing systems [3], [4], [5], [9]. Another approach consists in splitting the file and replicating the different blocks independently on the network. We propose an improvement of this method, consisting in splitting the file into k blocks, and then encoding them with an erasure code (these codes are classically used into FEC – Forward Error Correction – domain). The total number of blocks become n (where $n>k$) and these n blocks are disseminated in the network. This approach was firstly introduced in a fault-tolerant goal in distributed storage systems (see e.g. [10]), but this dissemination scheme drastically improves the average downloading time. This can be explained by the erasure code properties which induce that only a subset of the total set of blocks is mandatory to reconstitute the original information. Any block holds the same amount of information. All clients will then be statistically located closer to the minimal necessary amount of blocks among the erasure blocks copies than classical approaches. We show in this paper that an accurate tuning of dissemination parameters helps ensuring statistical content-access QoS in the P2P network.

The paper is structured as follows. The next Section presents the use of erasure codes to enhance data access performance. The performance of this proposition is studied in the Section 3. Section 4 introduces the statistical content access quality of service and proposes a simple case study in the context of media streaming. Section 5 deals with the proposition of a new fast MDS erasure code matching the constraints of the previously introduced P2P system. Concluding remarks are given in Section 6.

2. USING ERASURE CODES TO ENHANCE DATA ACCESS PERFORMANCES

2.1. Erasure Codes and Peer-to-Peer Context

The error correcting codes are a classical mechanism to protect information against errors or losses in transmissions. The principle is to add redundancy to the transmitted information, permitting receivers to recover the whole original data, even after experiencing transmission errors.

At higher layers of communication architectures, losses of data units are usually recovered by using Automatic Repeat ReQuest (ARQ) techniques. However, in several contexts, the retransmissions are replaced by erasure codes, often called Forward Error Correction (FEC) codes, which are error correcting codes designed for the erasure channel. Classically, an *[n,k]* erasure code considers a group of *k* packets and generates *n-k* redundant packets. Under these assumptions, the fundamental property of erasure codes allows the receiver to recover the k initial packets as soon as it receives any *k* packets among the *n* emitted ones. Note that this property holds only for maximum distance separable (MDS) codes such as Reed-Solomon-based codes [11], [12], [13]. Recently, a new generation of error-correcting codes with very fast encoding-decoding algorithms was re-discovered and proposed for the erasure channel [14][15], [16]. These codes have a linear coding and decoding complexity and become almost MDS when the length of the code n tends to infinity [16].

In networking area, erasure codes are used in various contexts. Most of the reliable multicast transmissions make use of erasure codes to avoid the ACK/NACK implosion due to a large number of acknowledgements. For real-time transmissions (e.g. video-conference), erasure codes avoid the delays due to the retransmissions of lost packets. In the area of distributed storage, instead of simply replicating data on several servers, fault-tolerant systems may use FEC techniques to improve their reliability. It is performed by encoding the data, splitting it up and distributing the fragments over various servers [17]. The basic idea is that a redundant fragment situated on a server can compensate for the loss of any other fragment due to a system failure. From a fault-tolerance point of view, the use of FEC reduces mean time of failures by many orders of magnitude, compared to replication systems with similar storage and bandwidth requirements [10].

A P2P system uses a peering architecture that offers the support for various P2P services such as applicative multicast communications or file sharing between peers. Examples of famous file sharing P2P systems are Napster [9], Gnutella [4], Kazaa [5] or Edonkey2000 [3]. Using these systems, each node is able to determine its peers and to localize data over the peer network. Furthermore, the peering algorithms allow data dissemination and cost determination in terms of bandwidth, for peer to peer data transfers. Considering data dissemination, [18] proposes specific algorithms to optimize erasure encoded data in the context of mirroring systems. Concerning the data localization among peers, several efficient techniques have already been proposed in [19].

P2P files sharing systems have specific characteristics compared to classical distributed storage ones. A major concern is the potentially large number of peers (e.g., up to 1.5 million users online at any time for Kazaa System in May 2002 [5]). Moreover, these peers strongly differ from storage systems due to the connection lifetime in the P2P network, the available throughput and the quantity of stored data. These parameters can vary between three and five orders of magnitude across the peers in the systems [6]. In this context, the design of decentralized systems which take into account all these properties is a current research issue.

Our proposal is particularly well suited to this heterogeneous and mobile context. Indeed, this paper focuses on the use of erasure codes to improve the content access Quality of Service by enhancing the data dissemination into the P2P networks. In addition to an enhancement of the global reliability of the data (similar to a distributed storage area [10]), this dissemination permits to improve the downloading performance for the end-users.

2.2. Dissemination Approaches in P2P Networks

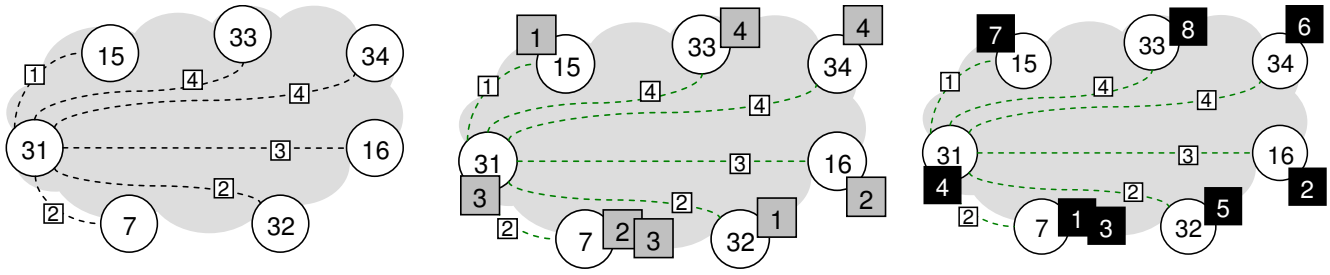
Some of the previously described P2P services are used by the three different dissemination approaches considered in this paper. Note that we do not make any supposition on the content or on the properties of the different peers. The only parameters taken into account are the total numbers of replicas in the network and the available bandwidth towards a considered peer.

The first content replication strategy consists in replicating the entire files in several peers on the network. These file replicas are supposed to be randomly disseminated in the network. The second approach splits the files into *k* blocks and replicates them independently in the network. The last approach [20][21] is based on the following method. When a file must be published into the network, it is firstly cut into *k* blocks. These various blocks are encoded using erasure codes. The *k* blocks constituting the initial shared file then become a set of *n* blocks. Depending on the coding technique, the first *k* blocks can be or not the *k* original blocks. In the first case (i.e., systematic form), they just result from the original file splitting. The *n-k* next blocks integrate the redundancy introduced by the erasure encoding. In the second case (i.e., non-systematic form), the original information contained into the *k* blocks is distributed over the *n* blocks. The last phase of publication is the dissemination stage of the various blocks over the P2P network.

Considering the various blocks disseminated into the network, downloading a complete file is equivalent to downloading any *k* blocks among the *n* ones. Hence, compared to classical approaches, there are greater choices for the sets of *k* blocks in such a way to reconstitute the original data. Note that availability and robustness of the system is also increased. When these blocks are disseminated over the P2P network, searching and throughput measurement services help to determine the closest ones by considering a certain cost function (e.g., the largest bandwidth). Then, the *k* closest blocks are downloaded in a classical way. The original data file can be finally obtained from the decoding of those *k* blocks.

2.3. A Simple Case Study for Sequential Downloading

To illustrate the concept and the benefits of the proposed scheme, we present a case study based on a (very simple) P2P network. In this case study, the file transfer between two peers in ensured by a direct connection with a given cost. Note that we do not consider the classical representation of P2P networks as a graph. The network structure is fixed and different blocks or file distribution schemes are compared according to their downloading cost, as shown in Figure 1.



(a) Entire File Replication *(b) Simple Block Replication* *(c) Coding Block Replication*

Figure 1 A simple case study to compute cost to recover entire file

We suppose the node 31 wants to recover the file. The first approach, named *Entire File Replication*, consists in keeping the entire file which is simply replicated into 2 random nodes of the network (e.g., node 32 and node 7). When split into blocks (see next approach), the considered file is supposed to be cut in 4 blocks of the same size. Then the cost to download the file is equivalent to download 4 blocks with a cost of 2, then, $C=4\times2=8$.

This approach can be compared to the *Simple Block Replication* that considers the same file split up into 4 blocks. In this case, the 4 different blocks are duplicated 2 times so that the total load is always 8 blocks. In

this case, 1 block is already present in node 31 (cost=0), 1 block needs a cost of 1, 1 block needs a cost of 2, and 1 block needs a cost of 4, then, C=1×0+1×1+1×2+1×4=7.

In the proposed *Coding Block Replication* approach, there is still *8* blocks disseminated as previously, but each of these blocks has been encoded with a MDS code. This encoding permits to choose any 4 over the 8 blocks of data, allowing a better choice of peers. As a result, the *Coding Block Replication* cost is: 1 block needs a cost of 0, 1 block needs a cost of 1, and 2 blocks need a cost of 2, then, C=1×0+1×1+2×2=5.

The following section presents a more general and realistic model to quantify the performance obtained by a P2P system implementing a *Coding Block Replication* approach.

3. DATA ACCESS PERFORMANCE EVALUATION

One major characteristic of recent P2P systems is to support a parallel access to several peers [3], [4] (i.e., several blocks are downloaded at the same time). Actually, when a file is located in several peers, several connections towards these peers can be used to download the different parts of the requested file. In this context, a sequential downloading scheme (i.e., each block is downloaded one after the other) could be considered as a particular case of parallel downloading scheme. The parallel strategy drastically improves the total downloading time. Those gains are evaluated in the context of mirror sites in [22].

3.1. Problem Modelling

We propose a simple model allowing computing the cost to get an entire file over the three different dissemination approaches. The cost function is associated to the time to get the file.

For the first strategy, (a) Entire File Replication, we consider that the *r* file replicas can be downloaded independently with the bandwidths $Bw_1, Bw_2, ..., Bw_r$. By opening *r* parallel connections towards the *r* peers and assuming that each connection downloads different parts of the file, it could be obtained a total bandwidth up to $Bw_1 + Bw_2 + ... + Bw_r$. This maximum is reached if all the connections are bottleneck-disjoint i.e., without any side-effect between the various connections. In a P2P network, where the connections can be established among several thousands different peers, it is very difficult to evaluate the presence of bottlenecks due to parallel-downloads of different parts of the same file. Thanks to the over-provisioning of the core network infrastructure, we consider that no bottleneck will appear in the backbone due to the traffic generated by these parallel downloads. This assumption is strongly confirmed by the campaign of measures presented in the next part of this section. However, we consider a bottleneck may be situated near the client, at access network, due to bandwidth limitation.

Let us denote this access bandwidth by Bw_0. By using this new parameter, we can state that the available bandwidth is now equal to $\min(\sum_{i=1}^{r} Bw_i, Bw_0)$. The cost to get the entire file is then:

$$(1)\ C_{(a)} = \frac{S_F}{\min(\sum_{i=1}^{r} Bw_i, Bw_0)}$$

Where S_F is the size of the file.

For the second strategy, (b) Simple Block Replication, the file is segmented into *k* blocks of size S_b, and each block is replicated *r* times in the network. Let us denote by $Bw_{i,j}$ the available bandwidth for the download of the i^{th} replica of the j^{th} block. This block can be downloaded for a cost equal to $S_b / Bw_{i,j_i}$ where $Bw_{i,j_i} = \max_{j=1..r}(Bw_{i,j})$. Note that parallel accesses to several replicas of the same block could be opened, but this would be equivalent to have a larger number of blocks and could also decrease performance due to the added load. So we only consider one connection per block. With this hypothesis, *k* parallel connections are

opened. The minimal cost to get the entire file is then $\max_{i=1..r}(S_b / Bw_{i,j_i})$.

As stated previously, this expression holds if there is no bottleneck due to the parallel downloads. Like in the previous case, we only consider the bottleneck due to the access bandwidth limitation, still denoted by Bw_0.

If $\sum_{i=1}^{k} Bw_{i,j_i} < Bw_0$, then the minimal cost is given by the previous formula. On the other hand, if $\sum_{i=1}^{k} Bw_{i,j_i} \geq Bw_0$, due to the TCP fairness, we consider the bandwidth Bw_0 is equally split into the k connections, i.e. a bandwidth of Bw_0 / k is supposed to be available for each connection. But if there is a bandwidth $Bw_{i,j_i} < Bw_0 / k$, the corresponding connection is not modified by the limitation resulting from this bandwidth limitation. The previously given cost value only depends on the minimum value of the Bw_{i,i_i}, then we can state that:

$$(2)\ C_{(b)} = \begin{cases} \dfrac{S_b}{\min_{i=1..k}(Bw_{i,j_i})} & if \ \min_{i=1..k}(Bw_{i,j_i}) < Bw_0 / k \\ \dfrac{S_b}{Bw_0 / k} & else \end{cases}$$

Note that in the second case $C_{(b)}$ is equal to S_f / Bw_0.

The case (c) Coding Block Replication can be analysed like the case (b). The k original blocks constituting the file become now n blocks due to code redundancy. Getting the file consists now in downloading the k blocks of minimum cost over the n blocks. If the n blocks are replicated r' times over the network, we define $Bw_{i,j_i} = \max_{j=1..r'}(Bw_{i,j})$ for $i=1,..n$ and the function $f : \{1,...,n\} \longrightarrow \{1,...,n\}$ such that:

$$(3)\ Bw_{f(1),j_{f(1)}} > Bw_{f(2),j_{f(2)}} > ... > Bw_{f(n),j_{f(n)}}$$

Then, the only difference with the case (b) is that the considered bandwidths are $Bw_{f(1),j_{f(1)}}, Bw_{f(2),j_{f(2)}}, ..., Bw_{f(k),j_{f(k)}}$. Using similar considerations than in the previous case, the cost is expressed as follows:

$$(4)\ C_{(c)} = \begin{cases} \dfrac{S_b}{Bw_{k,j_{f(k)}}} & if \ (Bw_{f(k),j_{f(k)}} < Bw_0 / k) \\ \dfrac{S_b}{Bw_0 / k} & else \end{cases}$$

Note that (4) uses the fact that $Bw_{f(k),j_{f(k)}} = \min_{i=1..k}(Bw_{f(i),j_{f(i)}})$.

3.2. Simulation Study

3.2.1. Implementation

The simple model given in the previous section has been implemented. The main idea is to consider *3* arrays of size *r* for entire file replication and of size *k.r* and *n.r'* for respectively Simple Block Replication and Coding Block Replication. To be fair with the previous approaches in terms of data storage load over the P2P network, we assume the total number of blocks disseminated into the network to be constant, then:

$$(5)\ r' = \frac{k \times r}{n}$$

The arrays are filled by random numbers following a certain probability law described in the next section.

Each value of the arrays represents the bandwidth available for downloading the corresponding block (or file for the first array). Equations given in the previous section allow an easy and fast computing of the download cost for the entire file. Various gain percentages are obtained by comparing the considered block replication approaches (i.e., Simple and/or Coding) to the basic Entire File Replication in a parallel downloading context. Each experience is done 1,000,000 times. Nevertheless, for the reference Entire File Replication approach, we limit the total number of parallel connections to the *k* best quality connections. We choose to apply such a restriction first to avoid opening too many TCP connections from receiving peer to remote peers, and secondly to be fair with the other downloading approaches that are not authorized to open more than k connections.

In all simulations except the last one, the access bandwidth at transport level is supposed to be 500Kb*/s*.

3.2.2. Probability Law

As explained in the previous section, the simulations are computed from bandwidths randomly generated using a particular probability law. Actually, this law must represent the peers' distribution in terms of bandwidth for a given receiver point of view. To have a realistic model, we made measure campaign on the Gnutella network in order to obtain a representative spreading of the different peers. This has been achieved both in the case of sequential and parallel downloading in order to validate the assumption made in 3.1: "bottleneck will not appear in the backbone due to the traffic generated by the parallel downloads". We made the measurement in a condition where access network was not a bottleneck bandwidth.

Using a Gnutella client, we measured the average bandwidths when downloading 1,000 different files between the 1st and the 29th of January 2003 on a total of 1,000 different peers nodes. Those 1000 files were downloaded 2 times. First, in the parallel measurement case, they were downloaded 10 by 10 on 10 randomly chosen different peers. We finally made 100 downloads on a total of 100×10 peers. Secondly, they were downloaded one after one for sequential measurement on the same set of peers used in the parallel case. Our receiving peer node was located on the University campus local area network, connected to a Gigabit Metropolitan network, this network being connected to the French research backbone RENATER network with a 155 Mbits/s access bandwidth. The downloaded files had a size varying between 3 and 4 MegaBytes (MB). Thanks to these large sizes, the observed bandwidths can be considered representative of the connection link between the considered peer and our P2P node. Moreover these sizes of the downloaded files are typical file size on these P2P networks.

From the observed bandwidths, we have computed the number of peers for each bandwidth interval of *5Kbits/s*. These results are presented in Figure 2.

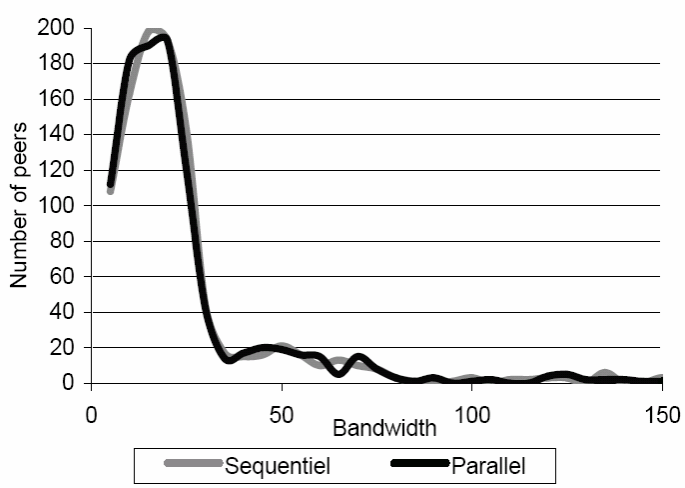


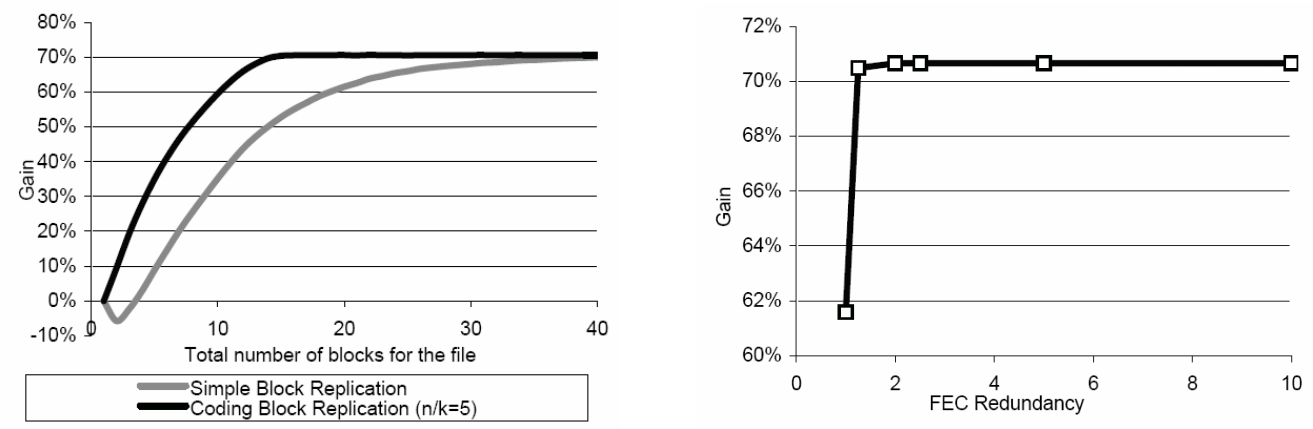
Figure 2 Topology of the Gnutella network in terms of bandwidth in Kbit/s

Figure 2 shows that the end-to-end bandwidth of most of the connected nodes is situated in an interval bounded by *1Kbits/s* and *30kbits/s* whatever the technique used (sequential or parallel). The similarity between the two curves confirms the assumption made in 3.1, i.e., no bottleneck exists due to parallel downloads.

3.2.3. Results

a) Impact of Number of File Blocks

The first simulation studies the impact of block splitting granularity, i.e., the number of blocks k constituting the file. Note that in this simulation, the total amount of data disseminated into the P2P network is always constant. In the Simple Block Replication approach, the number of replicas is r=10. In the coding approach, the redundancy factor n/k is equal to 5 and the number of replicas is r'=2 (see (5)). Figure 3-(a) shows that the coding approach gives always better performances than the Simple Block Replication approach. Moreover, the maximum gain (up to *60%*) is obtained very soon, *10* blocks seems to be a good compromise between the performance gain and the content dissemination level.



(a) (b)

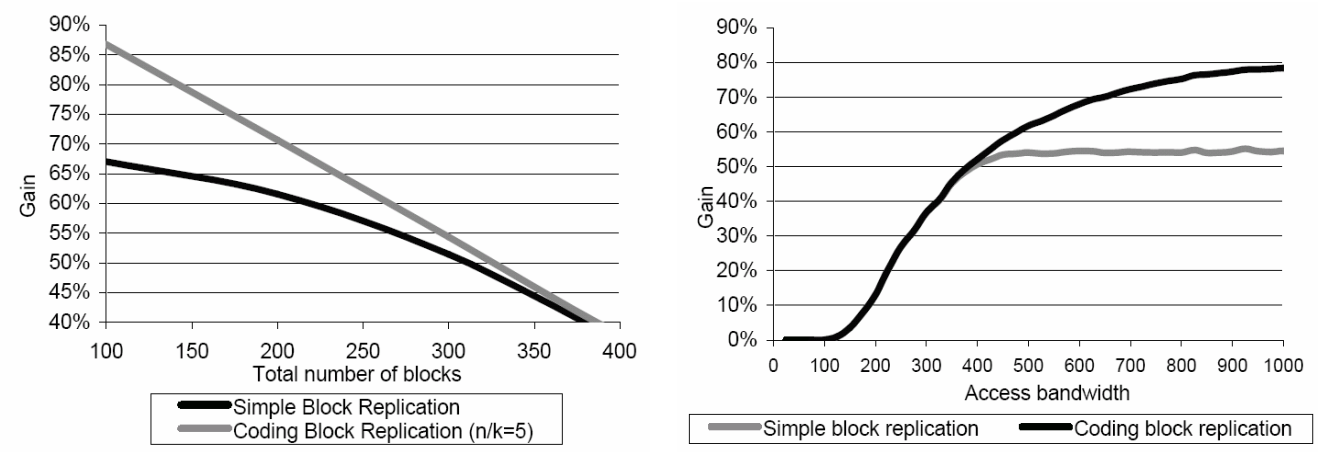
Figure 3 (a) Impact of the number of file blocks. (b) Impact on code redundancy factor

The negative gain for the Simple Block Replication approach is explained by the specific constraints defined in Section 3.2.1., limiting the number of parallel connexions to k (i.e., the number of blocks the file is split). This artifice is not completely fair when file is split into only few blocks, but ensures realism when the

number of disseminated blocks is large.

The impact of code redundancy factor is studied in the next simulation, where the number of file blocks is *k=20*, the number of replicas is *r=10* for the reference case (Entire File Replication). In order to verify (5) for the coding approach, the number of replicas *r'* is varying. Figure 3-(b) shows that a low redundancy factor *n/k* (*2* or *3*) is sufficient to obtain very important gain (up to *61%* in this case study).

The next simulation makes varying the number of replicas *r* and *r'* to study the impact of the total number of blocks. While the number of blocks *k* equals 20, the redundancy factor *n/k* equals 5 (then, *n*=100). Following (5), *k.r* is equal to *n.r'*. The impact of the total number of blocks *k.r* (Simple Block) or *n.r'*(Coding Block) is presented in Figure 4-(a).



(a) (b)

Figure 4 (a) Impact of total number of blocks. (b) Impact of access throughput (Kb/s)

The Figure clearly shows that the Coding Block Replication is more powerful with a limited total number of blocks (up to *350*). After this limit, the two curves converge. This convergence is explained by the fact that a large number of file or blocks replicas make them very accessible, then, coding approach is less interesting at *n/k* constant.

b) Impact of Access Bandwidth

The last simulation finally studies the impact of access bandwidth variation on performance. The file is made of *k*=20 blocks, each one replicated *r*=10 times for the Simple Block Replication approach. The Coding Block Replication approach uses a redundancy factor of *n/k*=5 and then a number of replicas *r'*=2. Figure 4.(b) presents the obtained results.

This curve particularly demonstrates the domain of interest of the coding approach. The left part shows that up to 350 *Kb/s*, there is no significant gain between simple and coding approaches. This result illustrates that no performance gain can be obtained when the bottleneck is situated on the access network. The development of high speed access using technologies such as xDSL makes realistic the situation where access bandwidth is high. The right part of the curve illustrates the gain of Coding Block Replication compared to the Simple Block Replication technique.

4. PROVIDING STATISTICAL QOS ACCESS GUARANTEES WITH REPLICATION

4.1. Quality of Service in Internet Networks

IP (Internet Protocol) protocol stood out to itself as the universal internetworking protocol for the transfer of computer data. The service provided by IP is "best effort", without any Quality of Service (QoS) guarantees. This type of service, without commitments and fair among the users, allowed a fast and flexible development

of the Internet infrastructure. However, QoS observed by end-user is not very stable. New time-constrained multimedia applications are very sensitive to these variations and require higher level of QoS guarantees. Various approaches, situated at network, transport or application level are proposed to enhance the QoS support in such networks.

The coverage of QoS in Internet network was traditionally situated at the transport layer, adapting the raw characteristics of the network services to the communication needs of the applications. Nevertheless, this service adaptation is limited to certain parameters such as reliability or order. Some applications ask for other service parameters, for example performances in term of end-to-end delay, throughput, etc. Another approach is to manage QoS at network level (and below). It consists to provide differentiated services to users, based on network resources management. Examples of such approaches are proposed in Intserv or Diffserv. However, various problems such as scalability and interoperability make complicated a global deployment of those techniques. On the other hand, managing and maintaining hard QoS guarantees on end-to-end delay, jitter or throughput can be considered too expensive for certain traditional applications like Web ones.

As a result, the global deployment of QoS support in the Internet is still not operated. An alternative solution has been developed to moderate the increasing raise of the traffic without over-provisioning the core networks too much. This solution consists in replicating contents in various points through the network. This technique is largely used in caching framework or CDN (Content Delivery Network). A set of distributed servers (cache, proxy-server, surrogate) stores a copy of content initially available on a centralized server. This copy can be then substituted to the original content as long as consistency is ensured. This approach leads to decreasing the traffic in the network backbone: the network backbone is only used for conveying the content to the set of distributed cache servers. Moreover, it also increases the number of users because of the load balancing between original server and caches. Finally, end-user quality of service is greatly enhanced because this approach reduces the networking resources involved between users and contents.

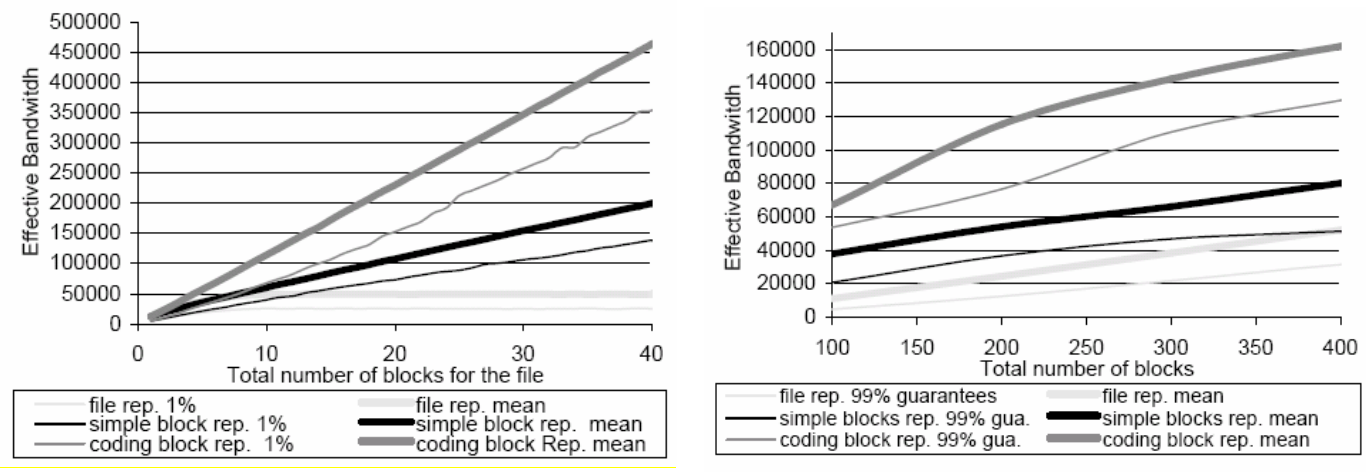
These solutions are a way to improve the content access QoS and thus to satisfy the user needs. The remaining of this paper study how those replication techniques in P2P environment can improve content access QoS.

4.2. Content Access QoS in P2P Networks

Quality of service in P2P network can be defined in term of content access availability. Studies on QoS in P2P networks mainly focuses on studying the files popularity in order to use heuristics schemes to duplicate data [23]. The QoS is associated to the reliability and the delay to retrieve a given shared data. According to the distribution of peers over the network, the type of statistical guarantee that can be obtained could be: "*Given the access bandwidth of a peer, the probability that it downloads a content C with an effective bandwidth higher than a given threshold is x".* This threshold will depend on the dissemination strategy, the number of replicas in the network, the number of file blocks k (second and third strategy) and the erasure encoding rate R (third strategy). By taking into account these parameters, we claim it will be possible to statistically ensure this type of guarantee.

To evaluate the statistical content access QoS guarantees, we compute the cost to get an entire file over the three different file distribution approaches, in a parallel downloading context. Using the model presented above, we can then compute for each replication technique, the time to download the entire file in a large P2P network of 10,000 peers. This operation is repeated a sufficient number of times (i.e., 10,000) to obtain accurate statistical measures: (1) the mean throughput, (2) the worst throughput for the subset of *99%* best peers. These last measures can be considered as statistical content-access QoS guarantees. In the simulation, the access bandwidth is supposed to be sufficiently high to avoid a bottleneck in the access link.

The goal of the first simulation is to study the impact of the number of blocks k constituting the file for the case of Simple Block and Coding Block Replications. In the simple block approach, the number of replicas is r=10. In the coding approach, the redundancy factor n/k is equal to 5 and the number of replicas is r'=2 (see equation (1)). As already shown in the previous performance study, the coding approach always give a better effective bandwidth and content access QoS guarantees, than the simple block and entire file approaches.



(a) (b)

Figure 5 (a) Impact of the number of file blocks on effective bandwidth variation in bit/s. (b) Impact of total number of blocks (replicas*erasure code redundancy) on effective bandwidth in bit/s.

Moreover, we can see that with a total number of blocks k equals to 20, the average and all the guaranteed bandwidths obtained with the coding approach are already higher than two times better than the simple blocks approach.

The objective of the second simulation is to evaluate the number of replicas r and r', on the effective bandwidth. Indeed, the more a file is replicated in the network, the more the content access bandwidth increases. While the number of blocks k is fixed to 20, the redundancy factor n/k equals 5 (then, n=100). Following equation (5), $k.r$ equals to $n.r'$. The impact of the total number of blocks $k.r$ (Simple Block Replication) or $n.r'$(Coding Block Replication) is presented in the 0-(b). Associated to the results given in Figure 4-(a), the coding approach is more powerful with a limited total number of blocks. Even if the effective bandwidth is still growing, the interest in using coding rather than simple block approach decreases as the number of blocks increases a lot.

4.3. Case Study: Streaming in P2P Networks

This section proposes to apply the previous results in the context of multimedia streaming. The objective is to ensure the playing of a multimedia stream in a P2P network, using content replication to obtain content access statistical QoS guarantees. The previous study helps determining a content replication pattern in the P2P network to ensure the required networking throughput for the stream.

Let us consider a streamed content encoded with a playing rate R_p. This content is supposed to be cut into a set of chunks of same size s, which are stored into the P2P network. Each chunk is considered as an input of the three dissemination techniques, i.e., it can just be replicated entirely over the peers (Entire File Dissemination) or cut into a number of replicated blocks (Simple Block Replication) or erasure encoded blocks (Coding Block Replication).

We suppose the first chunk to be played at date tp_0. The network throughput X_0 to get the first chunk is supposed fixed, defining the request time $tp_0 = t_0 + \frac{s}{X_0}$ (we suppose the playing does not begin before the first chunk joins completely the receiver). The second chunk, and all or a set of the remaining one's are supposed to be requested at the same time t_0. This yields to the decreasing throughputs requirements $X_1, X_2 \ldots, X_n$ for each chunk. A general expression of X_n can be defined as follows:

$$(6) \qquad X_n = \frac{s}{tp_0 + \frac{n.s}{R_p} - t_0}$$

This expression allows a rough estimation of the throughput requirement for each chunk allowing a correct playback rate. Considering the set of throughput computed by the last expression, the results given in the previous section will help determining the required parameters.

Let's take a practical example. Considering the playing of a large audio file of a total size 5MB and encoded at a rate of 128Kb/*s*. We propose to cut the file into 5 chunks of 1MB. We suppose to have an ADSL access with 512 Kb/*s* throughput and to devote 50Kb/*s* to download the first chunk. This leads to a waiting time of roughly 160s before the mp3 begins to be played.

The previous results (see Figure 5-b) are used to determine how each of the three dissemination techniques can provide sufficient guarantees to reach the throughput objective (*99%* guarantees). The first dissemination technique (Entire File Replication) could provide sufficient guarantees to reach *50Kb/s* in these conditions if the first chunk would be replicated much more than *400* times (see light grey fine line in Figure 5-b). We consider this approach too costly. Considering the constant number of block (k=20) and FEC factor (5) in each chunk, the two other approaches leads to the dissemination configuration proposed in the next Table.

Chunk Number	Needed Throughput Requirement (Kb/s)	Simple Block Replication (k=20)	Coding Block Replication (k=20, code redundancy Factor =5)
		r (size in KB)	r' (size KB)
1	50,0	20 (20000)	1 (5000)
2	28,1	8 (8000)	1 (5000)
3	23,0	7 (7000)	1 (5000)
4	19,5	5 (5000)	1 (5000)
5	16,9	5 (5000)	1 (5000)
Total	137,5	900 blocks (45000)	500 encoded blocks (25000)

Figure 6 Dissemination parameters for chunk throughput requirements

Few remarks can be deduced from the previous example:

- The needed maximum throughput to play the media is approximately equal to the playing rate (137,5Ko/s). This maximum is only reached at the beginning of the mp3 streaming and decreases during the playing phase.
- The needed number of streams to implement this example with Coding Block approach is 20×5=100 and decreases during playing.
- The Coding Block Replication approach is clearly more efficient than the Simple Block Replication approach in terms of data storage.
- For the same content access QoS, the storage load is roughly divided by 2.

5. IMPLEMENTATION USING A NEW FAST MDS ERASURE CODE

A critical issue in the implementation of the proposed coding approach concerns the erasure code. In this Section, the constraints related to the Coding Block Replication application are evaluated in such a way to determine the best suited class of erasure codes. In this context, we propose a new MDS erasure code based on Cauchy matrix over a prime field. This code has been implemented and coding and decoding performances are presented and compared to classical available implementations of comparable codes.

5.1. Coding Block Replication Constraints

The choice of an erasure code must be done, taking into account several parameters such as:

- The encoding and decoding running time;
- The efficiency, i.e. the quantity of necessary redundancy to recover a codeword;
- The number of potential redundant packets.

Firstly, let us describe the implementation of the erasure codes in this context. We consider that a file which must be shared onto the P2P network is first segmented into k blocks. Then, the encoding process generates n blocks (possibly including the k initial blocks) which are disseminated in the network (in one or more replicas). A peer willing to download the file locates the k (or $k.(1+\varepsilon)$ if the code is not MDS) closest blocks and then downloads them.

The size of the blocks depends on the size of the initial file but usually, it is at least equal to several hundred of Kilobytes. For the transmission, each block is cut into several packets of several hundreds of bytes. In the encoding phase, the first packet of each of the k blocks is used to generate the first packet of each of the n encoded blocks. The same operation is done for the i^{th} packet of each block, where i varies from 1 to the number of packets of the blocks.

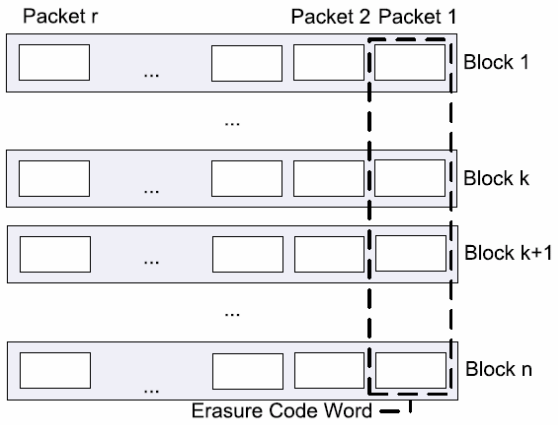


Figure 7 Encoding organization for pipeline processing.

This coding scheme has an interesting consequence on the decoding step. Indeed, we consider that the different encoded blocks are downloaded in parallel. The decoding of the codeword associated to the first packets can begin as soon as all the first packets are received. Therefore, the reception of the data and the decoding can be achieved in parallel with a pipeline approach. A strong constraint on the erasure code is to decode the data faster than the throughput of the transmission. In Figure 2, it can be observed that the end-to-end bandwidth between the different peers is generally low compared to the processing speed of usual codes. We can consider that all the different erasure codes satisfy this constraint, even for parallel downloads (see benchmarks of MDS codes given in Figure 9). Therefore, this constraint is not restrictive on the choice of the erasure code.

From the previous arguments, it appears that the available bandwidth is the bottleneck in the transmission-decoding pipeline. Hence, the system is optimized if the quantity of transmitted data is minimized. *This condition implies that the erasure code must need the minimum amount of data to recover the initial data*. Therefore the code must be MDS (see Section 2.1).

Another major point concerns the number of blocks useful for coding block approach. *The Figure 3(a) shows that data must be split in a small number of blocks (i.e. k)*. Note moreover that this number corresponds to the number of parallel downloads, i.e. to the connections simultaneously opened for a given file. Concerning the number of redundant blocks in the networks, it is clear that an optimal code would be able to minimize the number of replicated blocks in the network, i.e. to generate a sufficient number of different redundant blocks. Since the number of blocks in the network depends on criteria difficult to estimate a priori

such as file popularity, *the erasure code should be able to build the maximum number of different blocks*.

5.2. Description of the Erasure Code

The constraints described previously imply the erasure code to be MDS and to be able to generate a large number of redundant blocks. Moreover, it must have a decoding running time as short as possible.

For the packet erasure channel, a classical approach to obtain efficient erasure codes is to build them with a $k \times n$-generator matrix defined over a finite field, which is such that any $k \times k$-submatrix is invertible. The coding operation, presented in Figure 8, considers the k initial packets of size sz elements as a $sz \times k$-matrix and multiplies it by the $k \times n$-generator matrix to produce n packets of size sz. Upon the reception of k packets, the decoding algorithm extracts the corresponding $k \times k$-submatrix from the generator matrix and multiplies its inverse by the $sz \times k$-matrix corresponding to the received packets to recover the original ones. As a result, the two main points concerning the definition of an erasure code are the choices of the finite field and the structure of the generator matrix.

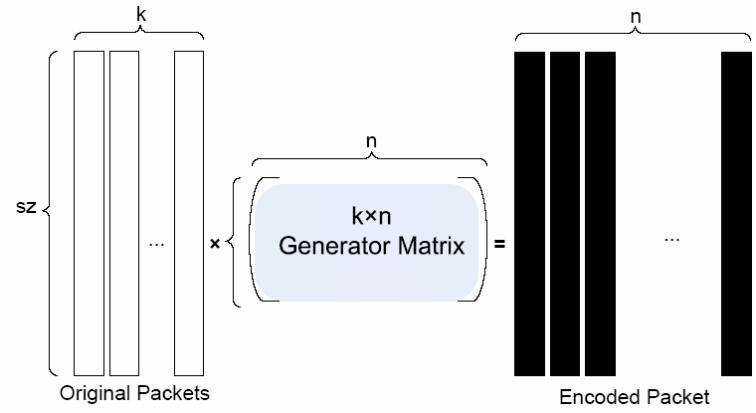


Figure 8 Coding operation.

Recall that finite fields are defined as a finite set of elements which has a field structure. Its cardinal number is necessary $p.m$, where p is a prime number. When m=1, the field is called a prime field and the elements of the finite field, denoted by *GF(p)*, can be represented by the integers $0,...,p$-1 where the addition and multiplication between two elements are done modulo p. When m>1, the field is called a polynomial finite field and is denoted by $GF(p^m)$. Its elements can be represented by the set of polynomials over *GF(p)* of degree less than m. The addition and the multiplication of two elements are done modulo an irreducible polynomial over *GF(p)* of degree m.

For the MDS erasure code, the choice of the finite field is important since (a) the maximal number of redundant blocks directly depends on the cardinal number of the field (see [11] chap. 11) and (b) the coding/decoding running times depends on the implementation of the operations of the finite field. Most implementations of erasure codes use a finite field $GF(2^m)$ where the binary polynomials of degree less than m are represented by m-bit vectors [12][13]. However, note that an erasure code over a prime field was proposed in [24], but its performance is lower than the one of [13].

For the choice of the generator matrix, two classes of matrices can be used. These two classes have the property to support fast matrix inversion (used for the decoding). Cauchy matrices can be used to directly construct systematic MDS generator matrix, i.e. which contains the $k \times k$-identity matrix on the k first columns (see [11]). Vandermonde matrices can also be used to build MDS generator matrix (see e.g. [12]) but not for systematic ones (see [25]). A construction of systematic generator matrix based on two Vandermonde matrices with optimal coding/decoding complexity is proposed in [25].

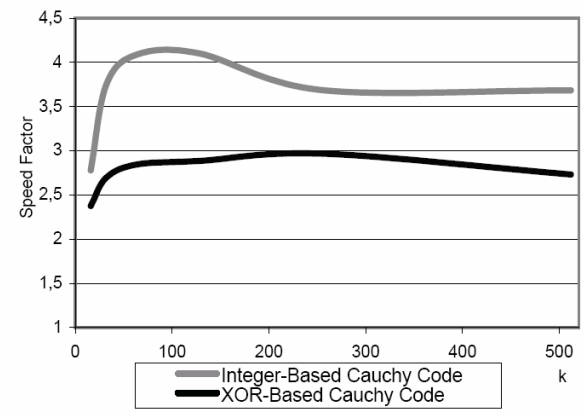
The erasure code proposed here is also based on Cauchy matrices. For the choice of the finite field, the basic idea is to use a large finite field where software multiplication and addition are fast. It appears that recent processors have these characteristics with integers. Then, it seems natural to use a prime field (denoted by *GF(p)*). However, two problems arise in such fields. First, data units are words of 2^m bits. Clearly, the set

of integers modulo 2^m is a ring, and not a field for $m>1$. Hence, the classical mathematical tools of finite fields can not be used. Then a correspondence between the computer words of a given length (e.g., m) and the p elements of the field must be defined (with $p \neq 2^m$). The second problem concerns the operations in the prime field which are done modulo a prime integer. The modulo operation is equivalent to a division and then is very costly in CPU cycles.

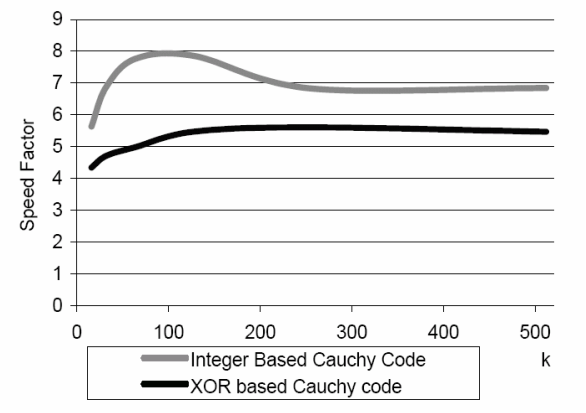
Like in [24] we use the field $GF(2^{16}+1)$. This value allows an easy correspondence between the field elements and the 16-bit words. Indeed, to represent the value 2^{16} which can appear in a redundant packet, we first look for an element of the field which does not appear in the packet (the size of the packet is necessarily less than the cardinal number of the field). Then, we add the value of this element (a 16-bit word) to the header of the packet and we replace all the occurrences of 2^{16} by this element in the packet. At the reception, the converse operation is done. The complexity of this treatment is linear and is negligible compared to the complexity of the encoding and the decoding phases. The problem of the modulo was reduced by observing that this operation can be done only once for several additions and multiplications (provided that the temporary results do not exceed the maximal value of integers). Practical simulations show that it is more interesting to use long computer words (64-bit) to store the temporary results and then to avoid modulo operations instead of using shorter words (32-bit). When 32-bit words are used (e.g. in the inversion of the matrix), we re-define the modulo $2^{16}+1$ operation by deducing it from the fast modulo 2^{16} operation.

The proposed code has been implemented. Figure 9 compares our proposed code with the different available implementations of erasure codes, i.e. a Vandermonde-based code [12] and a XOR-based Cauchy code [13]. Both are defined over a polynomial finite field. These simulations are done on a Pentium III 933 MHz running Linux. In these benchmarks, the length of the field elements is 16-bit for the three codes. The encoding and decoding speeds are given for k varying from 16 to 512 and for a value of n equal to $3.k$.

The next two Figures present the speed factor of coding and decoding operation of XOR-based and Integer-based compared to Vandermonde-based code.



(a)



(b)

	(a) Encoding speed (in MBytes/s)			(b) Decoding speed (in MBytes/s)		
k	**Int**	**XOR**	**VDM**	**Int**	**XOR**	**VDM**
16	10,10	8,63	3,63	8,94	6,89	1,59
32	5,88	4,23	1,57	5,56	3,81	0,81
64	3,18	2,21	0,78	2,63	1,69	0,34
128	1,39	0,98	0,34	1,10	0,77	0,14
256	0,51	0,41	0,14	0,48	0,39	0,07
512	0,28	0,21	0,08	0,21	0,17	0,03

***Figure 9** Performance comparison of (a) encoding and (b) decoding speeds compared to Vandermonde-based code.*

These comparisons show that the integer-based code runs faster than the other MDS codes. Compared to the Vandermonde-based code, the encoding (resp. decoding) speed is multiplied by a factor in the range from 6 to 8 (resp. 3 to 4). As indicated in [13], the XOR-based Cauchy code also runs faster than the Vandermonde code, but it remains lower than the integer-based code. The multiplicative factor is roughly in the range of 1.15 to 1.50 for the encoding and the decoding. Note that the type of the generator matrix is not related to the performance results.

Finally, the proposed code matches all the constraints of the coding block peer to peer application, optimal recovering capability, small number of initial blocks, potentially large number of redundant blocks and high processing performances. Moreover, this code would be appropriate in the general context of distributed storage applications [14].

6. CONCLUDING REMARKS

In this paper, we have presented a P2P file sharing system that uses an erasure coding scheme to disseminate information over the peers. Performance gains were estimated by comparing our system to classical approaches of various P2P systems. Several lessons emerge from the obtained results. First, the use of erasure codes allows increasing the average availability of data and the access bandwidth usage. Secondly, although the obtained results depend on the distribution law of the bandwidth over the peer network, there is a set of values (number of blocks, code redundancy factor, and number of replicas) that optimize the content-access QoS in the peering architecture. Optimizing these parameters can offer statistical quality of service guarantees in a Peer-to-Peer architecture. Similarly, given some bandwidth constraints (e.g. for video streaming) for a file, it is possible to determine the parameters (level of replication, number of blocks, code redundancy factor) that satisfy the constraints with a high probability. Finally, as our scheme is based on the use of an erasure code, we propose a new fast MDS erasure code matching the overall constraints of our approach.

ACKNOWLEDGMENTS

The authors thank Fabrice Frances for his help on the erasure code implementation.

REFERENCES

[1] Li Gong, “Peer-to-Peer Networks in Action”, IEEE Internet Computing, Vol. 6, N°1, January/February 2002.
[2] G. Coulouris, J. Dollimore and T. Kindberg. Distributed Systems: Concept and Design, Third Edition, Addison-Wesley, 2001.
[3] Edonkey2000, http://www.edonkey2000.com, May 2002.
[4] Gnutella Peer-to-Peer system, http://www.gnutella.com, May 2002.
[5] Kazaa Peer-to-Peer system, http://www.kazaa.com, May 2002.
[6] S. Saroiu, P. Krishna Gummadi, S. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems”, in TR UW-CSE-01-06-02, Department of Computer Science & Engineering, University of Washington, Seattle, WA 98195-2350, July 2001.
[7] K. Aberer, M. Punceva, M. Hauswirth, R. Schmidt, “Improving Data Access in P2P Systems”, in IEEE Internet Computing Volume 6, Issue 1, pp. 58-67, January 2002.
[8] E. Cohen, S. Shenker “Replication Strategies in Unstructured Peer-to-Peer Networks”, in Proceedings of ACM SIGCOMM, August 2002.

[9] Napster Peer-to-Peer system, http://www.napster.com, May 2002.
[10] H.Weatherspoon, J. D. Kubiatowicz, “Erasure Coding vs. Replication : A Quantitative Comparison”, in the First International Workshop on Peer-to-Peer Systems, LNCS, Pages 328-338, 2002.
[11] F. J. MacWilliams, N. J. A. Sloane, “The Theory of Error Correcting Codes”, North Holland, Amsterdam, 1977.
[12] L. Rizzo, “Effective Erasure Codes for Reliable Computer Communication Protocols”, in ACM SIGCOMM Computer Communication Review, Volume 27, Issue 2, Pages 24 – 36, April 1997.
[13] J. Bloemer, M. Kalfane, M. Karpinski, R. Karp, M. Luby, D. Zuckerman, “An XOR-Based Erasure-Resilient Coding Scheme”, in International Computer Science Institute Technical Report, TR-95-048, August 1995.
[14] J. S. Plank and M. G. Thomason, “A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide-Area Storage Applications”, in DSN-2004, The International Conference on Dependable Systems and Networks, Florence, Italy, June, 2004.
[15] R. G. Gallager, “Low-density parity check codes”, MIT Press, 1963.
[16] J. W. Byers, M. Luby, M. Mitzenmacher, A. Rege, “A Digital Fountain Approach to Reliable Distribution of Bulk Data”, in ACM SIGCOMM '98, September 2-4, 1998.
[17] M.O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance", in the Journal of the ACM (JACM), Volume 36, Issue 2, pp. 335-348, April 1989.
[18] M. Naor and R. M. Roth, “Optimal file sharing in distributed networks”, in SIAM Journal on Computing, Volume 24, Issue 1, pages 158-183 February 1995.
[19] S.Waterhouse, D. M. Dooling, G. Kan, Y. Faybishenko , “Distributed Search in P2P Networks”, in IEEE Internet Computing, Volume 6, issue 1, Pages 68-72, January 2002.
[20] L. Dairaine, L. Lancérica, J. Lacan, "Enhancing Peer-to-Peer Parallel Data Access with PeerFecT", in Proceedings of Fifth International Workshop on Networked Group Communications (NGC’03), LNCS 2816 Springer, Munich, Germany, September 2003.
[21] J. Lacan, L. Lancerica, L. Dairaine, “When FEC speed up data access in Peer-to-Peer Network”, in proceedings of ACM IDMS-PROMS, LNCS, November 2002.
[22] P. Rodriguez, A. Kirpal, E. W. Biersack, “Parallel-Access for Mirror Sites in the Internet”, In Proceedings of IEEE Infocom'2000, Tel-Aviv, Israel. March 2000.
[23] I. Clarke, O. Sandberg, B. Wiley, T.W. Hong, “Freenet: A distributed anonymous information storage and retrieval system", Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, pp. 311—320, 2000.
[24] C. Leicher, “Hierarchical Encoding of MPEG Sequences Using Priority Encoding Transmission (PET)”, International Computer Science Institute, TR-94-058, Nov. 1994.
[25] J. Lacan and J. Fimes “Systematic MDS Erasure Codes based on Vandermonde Matrices”, in IEEE Communications Letters, August 2004.

BIOGRAPHY

Laurent Dairaine is currently associate Professor at “École Nationale Supérieure d’Ingénieurs de Constructions Aéronautiques“ (ENSICA) in Toulouse, France and do his research in collaboration with LAAS-CNRS and TéSA research labs. He received a Ph.D. in computer science from Pierre et Marie Curie University of Paris in 1994 (LIP6 lab). In 1994 and 1995, he spent a year as visiting researcher at University of technology, Sydney (UTS), Australia. His research interests concerns quality of service and end-to-end communication architectures. He recently worked on application and transport protocols for IP Multicast over satellite, quality of service support over peer to peer and replication systems, and network emulation.

Jérôme Lacan received his M. E. and Ph. D. degrees in Computer Science from University Paul Sabatier, Toulouse, France in 1994 and 1997. In 1996-2000, he was assistant professor in University of Franche-Comté, France in Computer Science. Since 2001, he is with the Department of Applied Mathematics and Information Technology of the ENSICA and with the Cooperative Laboratory in Telecommunications for Space and
Aeronautics (TéSA) in Toulouse, France. His research interests are coding theory and network protocols.

Laurent Lancerica received his Ph.D. in Computer Science from Institut National Polytechnique of Toulouse in 2004. Since 2000 he was Ph.D. student and did his research in collaboration with ENSICA in Toulouse and TéSA research labs. His research interests are quality of service, overlay replication structures, Peer-to-Peer network and multimedia applications.

Jérôme Fimes has been a Ph.D. student since 2002 working with ENSICA, Toulouse France from which he passed an engineer degree in 2002, the Télécommunication Spatiales & Aéronautiques" (TéSA, Toulouse, France) research lab and Alcatel Space (France). His research interests mainly focus on the coding theory, the network protocols and the multicast reliability.